Overview

Apricot
Voice Toolkit

Published in the U.K. by

ACT (UK) Ltd
Shenstone House
Dudley Road
Halesowen
West Midlands B63 3NT

Published in the USA by

Apricot Inc.
3375 Scott Boulevard
Suite 342
Santa Clara
California 95051

It is important to point out that there is a high degree of variability in the performance of the speaker/computer combination across different speakers. This difference is invariably related to the speakers consistency and repeatability in the use of human language.

After a few attempts at training the VIS, the user/programmer will usually find that the recognition will improve quite dramatically.

Apricot Voice Toolkit : Overview

Contents

Introduction

One of the features integrated into the Portable microcomputer is a sophisticated speech recognition system.

The Apricot Portable is the first business microcomputer to integrate this function totally within the product. It provides the user with a much more natural way of communicating with the computer.

Instead of using just using the keyboard and optional mouse to enter information and commands into the Portable, the user can configure the computer to respond to spoken words and phrases.

The Voice Input System (VIS) is ACT's implementation of speech recognition. The VIS is an isolated word, speaker dependent speech recognition system. It allows the user to train in a vocabulary of isolated words or phrases into the computer, and then use these within an application or at the operating system level to command the computer to perform specific actions.

The function of this series of booklets is to show an applications programmer how to sucessfully utilise the Apricot VIS and how to integrate it within an application.

They should also be used by a user or a software dealer to configure applications to respond to speech input.

This booklet describes the function and capabilities of VIS in more detail, introducing it's two basic modes of operation, and also describes how to obtain best results when using the system.

A second booklet, "The VTRAIN Utility", illustrates how to generate and train the computer with a vocabulary.

The third booklet details the communication interface provided to integrate the speech recognition facilities into an application.

The interface has been specifically designed to keep the complexity of the speech recognition system in the background and make it extremely easy for an application to link into the VIS.

Word Differentiation

It is important when selecting the words that make up a vocabulary that it does not contain highly confused word pairs (such as "no" and "go"). ACT has provided the necessary software tools within the training utility to enable the user/programmer to measure the "recognisability" of a vocabulary, and thus isolate confused word pairs.

On finding a confused word pair, the user/programmer must then change one of the words to one which sounds different but has a similar meaning.

Training/Re-training

All users wishing to communicate with the computer using voice, must train the machine to recognise his speaking idiosyncrasies.

It does not mean that the user has to speak in a machine-oriented artificial voice. The main criteria is that the user speaks in the same clear and consistent manner both during the training routine and during operation within the application.

During the recognition phase of the training program the user will probably unconsciously speak to the machine in a clear and concise manner which is not his normal way of speaking. The user will tend to pronounce each word or phrase carefully and distinctly in an "unnatural" way.

When he is using VIS within an application it is likely that he will have the tendancy to speak less carefully. This will lead to misrecognition of certain words and phrases and will thus require a certain amount of re-training to gain the required "naturalness".

There is thus a need to both train and re-train the machine with a vocabulary.

Training is performed as a one off operation prior to initial installation of the application software, using the training program. Re-training can easily be done by identifying the problem words and re-invoking the training program, as required.

Using the VIS

To integrate a fully functional speech recognition system into a microcomputer and specifically when running it within an application, you must first appreciate some of the problems associated with this form of communication.

These can be broadly compounded into the following categories:

1. The Environment.
2. Word Differentiation.
3. Training/re-training.

Environment

VIS experiences the same problems as person to person communications. If a person cannot communicate with another person due to background noise, language/dialect unfamiliarity or other reason, then it is unlikely that VIS will fare any better.

Probably the most important problem in the office environment, where the majority of Apricot products are used, is that of background noise (that includes other speakers).

To combat this problem the user must:

1. Hold the microphone close to his mouth and specify during the training utility that it is a noisy environment.
2. Train the system with approximately the same noise level as there will be during recognition.
3. Use the mouse or keyboard instead of voice when the noise level is too high. (The VIS can be turned on and off by using the VOICE key and the SHIFT key together).

As ACT makes improvements in the VIS, it will become more tolerant to background noises and thus alleviate the above environmental problem.

The Apricot Voice Input System

There are two different methods of using VIS:

1. **Where the application has no knowledge of voice input**, and thus programs like Supercalc, WordStar, and even the operating system, can utilise the speech input capability without any alteration.

An example of this method is where a number of the WordStar commands (e.g. BLOCK, ON-SCREEN, QUICK, etc) are implemented for voice input.

Normally an application would configure function keys on the keyboard to activate these commands, and this method is therefore called the *Voice Function Key* mode of using the VIS. The user is able to speak these commands at any time during the execution of Wordstar and activate the relevant operation as though the key had been pressed.

This method can be applied to any application which uses the keyboard as its standard method of input.

2. **Where the application has a complete knowledge of the VIS**.

An example of this second method is where an application interacts with the VIS to provide a highly interactive speech recognition interface (such as found in the ACT application, Diary, which is supplied as a standard piece of software with the Apricot Portable).

The integration of speech recognition facilities into the application is the method by which best results are obtained. To facilitate this integration, we have provided a set of routines which hides the complexity of speech recognition, and allows an applications program to utilise the functions of VIS to the fullest.

This method of using the VIS is termed the *Voice Driven Application* mode.

Advantages of Using VIS

The use of speech as an input to the Portable has obviously a number of advantages over the more normal data and command input methods of the keyboard and mouse.

1. There is no need for the user to adopt "hand-to-eye" coordinated movements to search for a key or button on the keyboard or mouse.
2. The user does not have to continuously look from keyboard to screen to view the resultant action of a keystroke.
3. Speech is a far more natural method of communication than any electro-mechanical device.

VIS is not a total replacement for keyboard or mouse input, but is an adjunct to them. ACT has fully integrated VIS into the MS-DOS environment, in such a way that recognition can be easily linked into an application (i.e. as a loadable device driver).

Voice Function Keys

What the Voice Function Key mode allows the user to do is to configure an existing application or even the operating system environment to respond to voice input.

The way this is implemented is by using a simple utility (VTRAIN). This allows the user, a software distributor, or dealer to utilise the VIS for voice function keys.

This utility enables the creation of a vocabulary, speaker training, and the linking of the vocabulary into the operating system. In the Voice Function key mode, the VIS limits the user to using a maximum vocabulary of 32 words.

The way the utility works is as detailed in the following paragraphs.

Using SuperCalc as an example, the first operation the user has to do is create a vocabulary file, (e.g SCALC.VOC for the SuperCalc example).

The user then enters the text strings associated with the particular command function within the application (e.g. HELP, EDIT, COPY etc) into the file. In addition he needs to enter the 'key-code' associated with each command (e.g. for EDIT this would be */E*). This code will be placed into the keyboard buffer every time the user says the word EDIT which will then activate the relevant SuperCalc function.

One of the application tools provided by ACT is the training utility (VTRAIN) as used by the user in Voice Function Key mode. It enables the application programmer to build up vocabulary files and also test the recognisability of words within the files.

The limiting factors are the constraints of machine memory size and acceptable performance. (The training utility allows up to 9,999 words in a single vocabulary file).

This does not mean that the application has an unlimited pool of words active at the same time. It only can have a maximum of 63 words active at any one time, but can switch other words and vocabularies (up to 30) in and out to suit the context of the application.

Limiting of words in this way to within a defined situation is similar to what happens in person to person communications where we talk within the context of one subject or another. The conversation, like a particular command sequence in an application, generally only uses/requires a limited vocabulary.

The Voice Driven Application should always supply the user with an untrained vocabulary file(s). It is also possible that a pre-trained vocabulary can be supplied instead, but as the users' pronunciation will probably bear little resemblance to the original, this is currently of limited usefulness. Both types will require the user to invoke the training utility (VTRAIN) to train/re-train part or all of the vocabulary.

As the Voice Driven Application is fully aware of the VIS and can communicate directly with it, the vocabulary loader VOICE is used in a different manner. The format for loading a vocabulary file is similar to the Voice Function Key mode, it invokes the VIS as before, but the last parameter has a different meaning.

Instead of being the number of words within the vocabulary, it signifies the size of memory (number of buffers) allocated in memory for storing word models at any one time. This number is specified by the application writer and is related to the size of memory available on the machine.

The format of the vocabulary loader for a Voice Driven Application is as follows:

A>VOICE <FILENAME> <BUFFER SIZE>

This command specifies the first vocabulary file to be loaded into the system (and would normally be hidden from the user in a batch file), and also invokes the VIS, as with ACT Diary. The application then can switch in other vocabulary files, mask out individual words within a file, etc, by direct communication with the VIS to suit the particular context of the application.

To enable the VIS to produce best results, the user also has to tailor the system by activating a routine in the VTRAIN utility to answer the following questions:

1. Sex - Male, Female or Child; this helps the VIS to tailor its parameters to match the pitch of the users voice.
2. Background noise - Quiet, normal office or noisy; enables the VIS to set the microphone amplifier to the correct level.

The final routine of the utility is the speaker training aspect. This enables the user to train the system to recognise his pronunciation for each of the words used for the voice function keys.

The training utility asks the user to repeat the word a number of times to allow the VIS to take into account small variations in his pronunciation.

An optional routine allows the user to test the capability of the VIS to recognise the words selected prior to using the trained vocabulary within the application.

It is important that commands associated with each voice function key are not assigned to irreversable operations, e.g. deleting anything would be irreversable (unless the application provides an undo function).

When the user wants to use the voice function keys with an application, he has to load the vocabulary file prior to loading the application.

The vocabulary loader performs the following functions:

1. Enables the VIS and the VOICE key on the keyboard - initialised to the 'off' state.
2. Loads the trained word models (profiles) into memory and informs the VIS of their presence.
3. Loads the key-codes associated with the word models into memory.

The format used for loading the vocabulary file is illustrated by the following example:

A>VOICE SCALC 32

VOICE is the name of vocabulary loader, SCALC(.VOC) is the vocabulary file, 32 indicates the size of the vocabulary.

The same technique can be used to load any vocabulary file.

The maximum number of words that can be loaded is 32, regardless of the actual number of words in the file. (This will always be the first 32 words in the vocabulary file).

The user then loads the application and when ready, activates the VIS by pressing the VOICE key (F5) together with SHIFT. The VIS is now ready to accept any of the voice function key words that it has been trained to recognise.

The VIS can be switched off anytime during the application by repeating the keystroke combination, SHIFT + F5.

The user also has a method of unloading the VIS from the system on leaving the application which used voice input. This is again using the Voice command but specifying a different parameter. By typing:

Voice No

after leaving the application, the user can run another application without having to re-boot the system.

Working with Voice Function Keys

As the VIS is not 100% perfect at its job it will sometimes misrecognise a word or not hear it. In the first instance the application would perform the wrong function and in the second the user would have to say the word again.

When one voice function key is confused with another the application could perform a dangerous operation. It is therefore IMPORTANT that irreversible operations are not driven by voice function keys - e.g DELETE is normally irreversible. This problem occurs as the user does not have to confirm that the word the VIS recognises is the one he said.

With the voice driven application as described in the next section, this is not a problem as the application can be tailored so that it does not perform the required operation until the user confirms it. This is exactly the same mechanism the user has to do in a command driven system, where carriage return signifies "action the command".

Voice Driven Applications

A Voice Driven Application is one where the application interacts directly with the features of the VIS. The ACT Diary is such an application and is provided as standard software on the Portable.

The major difference between the Voice Function Key mode and the Voice Driven Application mode, is that in the application mode, the application is totally aware of the VIS. This makes it a great deal more flexible in terms of features and functions.

The application programmer is able to provide the user with a much more interactive and friendly interface which is also tolerant to misrecognition by the VIS, and as a result less prone to operational errors.

If the application fails to recognise the word, it will either ask the user to repeat the word, or ignore it completely. If the application is commanded to do a critical operation, which if not meant by the user could result in a catastrophic loss of information, the command is not actioned until the user has confirmed his intention.

The application mode sequence of speech input usually asks for confirmation before any action is taken. That is to say, the application can even enable the user to build up full sentences or phrases on the screen and then make him confirm it by saying (for example) "OK" before any action is performed by the application.

For example, in the ACT Diary, the user can say

"OPEN-AT" pause "NEXT" pause
"FRIDAY" pause "MORNING" pause "OK".

to open the electronic diary at the entry for next Friday morning.

This sort of verbose input allows the user to interact with his machine in a more "human" manner.

Most of the features described above cannot be implemented in the Voice Function Key mode where the application does not even know of the presence of the VIS.

The size of the vocabulary available in the voice driven application mode is also much more flexible than the Voice Function Key mode. Instead of being restricted to 32 entries, it is in theory "unlimited".

The VTRAIN Utility

Apricot
Voice Toolkit

Published in the U.K. by

ACT (UK) Ltd
Shenstone House
Dudley Road
Halesowen
West Midlands B63 3NT

Published in the USA by

Apricot Inc.
3375 Scott Boulevard
Suite 342
Santa Clara
California 95051

Once one of the command options has been selected, one other command option is provided on a pop across menu. This is the usual review feature.

Selecting *REVIEW* lets you review another page of entries. If this command is selected you can move from page to page either by using the mouse buttons (left for up/right for down) or ENTER/SHIFT + ENTER on the keyboard.

The sequence of operations required to test words in a vocabulary is as follows:

1. Either choose the

   SELECT ONE option and then select the line numbers of all the words you want to test (You can do this by either typing in the line number and pressing carriage return or by using the mouse). The maximum number of words you can select in a single session is 63 and a selected entry is identified within an *.

   or the

   SELECT ALL option which will automatically select the first 63 words in the file. (You cannot use this option for any entries with line numbers greater than 63).

2. Select FINISH and then select RECOGNISE to start the recognition program.
3. Say the words you marked for testing in any order, and as many times you wish. Each time you say a word, the program will respond with the PROMPT of the word it "thought" you said.

If you have marked entries on different pages and wish to see which entries you have marked, you can use the REVIEW feature without leaving the RECOGNISE phase.

As with all the other menus you can exit to the MAIN program by selecting FINISH twice. If you make a mistake with your selection, simply repeat the operation.

HELP displays a help text explaining the use of the testing operation.

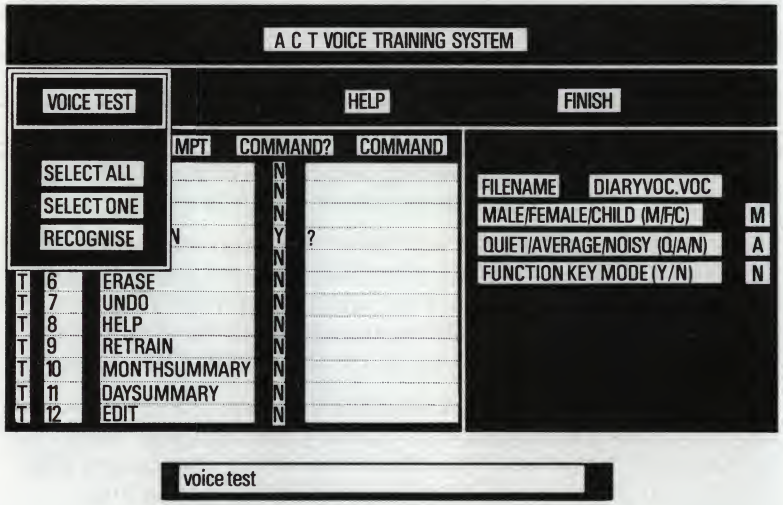
Testing a Vocabulary

The VOICE TEST command selection allows you to mark entries within a vocabulary file and then speak the marked entries to see whether the VIS correctly recognises the pre-trained PROMPT.

This can be used:

1. To test the accuracy of the VIS in interpreting your vocabulary correctly.
2. To isolate any confused "word pairs" if you've designed your own vocabulary.

Select the VOICE TEST option from the MAIN menu. The new menu now appears down the left-hand of the screen as shown below.



This provides you with the following command options:

SELECT ONE allows you to select a single entry in the vocabulary file for testing.

SELECT ALL allows you to select the first 63 entries in the vocabulary file for testing.

RECOGNISE allows you to test the ability of the VIS to recognise the marked entries in the vocabulary file.

Apricot Voice Toolkit : The Vtrain Utility

Contents

Introduction

This booklet describes the training utility software supplied with the Apricot VIS. The booklet is written in a simple tutorial style format. It is presented as a first time guide to someone who is not familiar with the training system.

The training utility serves a number of functions:

1. It enables the user to create a vocabulary file and train the recognition system to respond to his voice in the Voice Function Key mode.
2. It enables the user to train the system to respond to words from vocabulary files supplied by an application writer in the Voice Driven Application mode.
3. It also acts as an application tool for the application writer, enabling the writer to create vocabulary files and test the accuracy of the selected vocabulary.

The utility program is loaded using the TRAIN.BAT batch file.

If you wish to train only one or a few words in the vocabulary, you would select the TRAIN ONE option, and then enter the line-number of the PROMPT to be trained followed by carriage return.

A prompt will then appear on the screen as part of the following message:

Please Say <PROMPT>

Say the word clearly into the microphone in your normal way of speaking, each time the message is displayed. The system will then ask you to repeat the word five times.

After training the word, the system will report back to you whether it rejected the word. It will do this if it has had difficulty in interpreting the word. You should always re-initiate the training session on the currently selected word if you get more than one rejection, by pressing the REPEAT function key.

Press the carriage return key and the program will ask you whether you want to train another word.

If you do, specify the next required line number and repeat the same sequence of operations.

If you have finished the training sequence, press FINISH to return you to your entry point into the TRAIN menu.

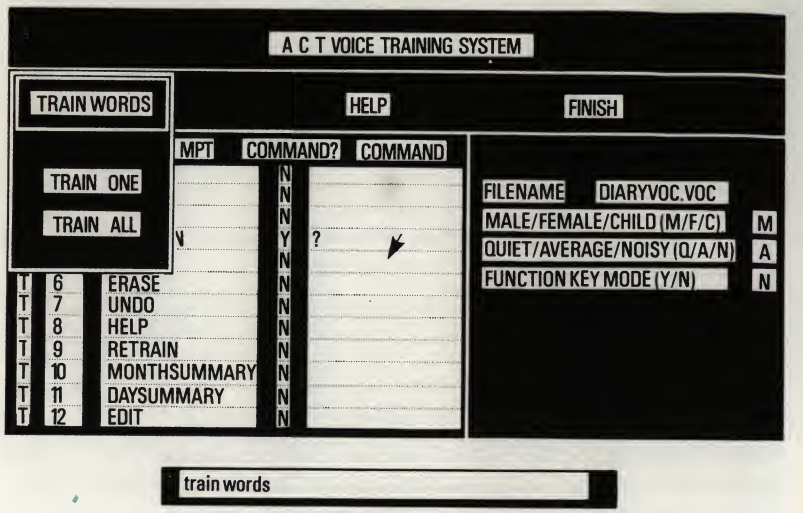
If you wish to train all the words in the vocabulary, select the option TRAIN ALL. The program will then ask you to say all the words in the vocabulary file starting from WORD 1 PROMPT. Any problems the system has in interpreting the word is reported back to you as for the TRAIN ONE option.

After training a word, pressing carriage return automatically increments the line number to specify the next line. This process will continue throughout the whole vocabulary file, unless you prematurely stop the sequence by selecting FINISH.

As with TRAIN ONE, you have to reply to the PROMPT five times. Say the word clearly into the microphone in your normal way of speaking, each time the message is displayed.

When you have trained all the words as required, select FINISH to return to the TRAIN menu, and select FINISH again to exit into the MAIN menu.

Following a training session you should generally test the accuracy of the VIS to recognise your vocabulary correctly by selecting the VOICE TEST command fron the MAIN menu.



This provides you with the following command options:

TRAIN ONE allows you to train single entries within a vocabulary.

TRAIN ALL allows you to train a full vocabulary in numerical sequence.

Once one of the command options has been selected, one other command option is provided on a pop across menu. This is a review feature. Selecting *REVIEW* lets you review another page of entries and is in effect a page up/page down command. If this command is selected you move from page to page either by using the mouse buttons (left for up/right for down) or ENTER/SHIFT + ENTER on the keyboard.

The training option allows you to train either the whole vocabulary or individual words.

Untrained words are automatically marked with a "*U*" (in the column nearest to the left-hand side of the screen) during the editing program when a new entry is added or an existing entry is changed.

This is altered to "*T*" (for Trained) after training the word.

Preliminary Details

Vtrain Overview

If you are using voice function key mode you will have drawn up a list of the spoken words and the corresponding control codes to define the appropriate action.

If you are using a particular application which can be used with VIS, the documentation with the application will tell you the appropriate vocabulary file(s) which have to be trained (e.g. for the ACT Diary application, this is DIARYVOC.VOC. There is also a vocabulary file which even allows you to operate the training utility by voice commands. This is called vtrain.voc).

If you are an application programmer who wishes to build up a vocabulary file or files, you will have drawn up a list of commands, filename(s), control codes and words to be used within your application design.

A vocabulary filename is always suffixed by ".VOC", and it is therefore advisable not to use this suffix for any other type of file.

What the voice training program does is create word-profiles of particular words or phrases that are spoken to the computer. It stores them along with the actions that the computer is to take when these word-profiles are recognised by the VIS.

The word-profiles, and information about how the computer handles them, are stored in the vocabulary file. There may be a number of vocabulary files, typically a minimum of one for each application that uses voice input.

The voice training program controls all aspects of the creation, amendment and deletion and training of complete vocabulary files or individual vocabulary records (word-profiles plus command information).

Once familiar with the function, you will find it simple to operate, as it is completely menu-driven. Each time you run the voice training program you are able to create, update or train/re-train any vocabulary file. Before using the program, it is important to know a little about the make-up of these files.

Vocabulary Files

The file has two parts, being the parameter information and vocabulary records.

The parameter information tells the voice recognition program about the expected environment that this vocabulary is to be used in; for example the sex of the speaker and the amount of background noise.

The vocabulary records contain the word-profile, the command or text to be sent to the computer when the word or phrase is spoken, and a prompt for repeating the word in the training session. The prompt is just the text displayed for each vocabulary record, and is our name for the action performed.

For example, if we wish the VIS to be used within the MS-DOS environment and to respond to the word "DIRECTORY" to provide a list of all the files on the disk, the prompt used would simply be the word DIRECTORY, and the command would be "DIR".

When the recognition program hears the word, it sends "DIR" (the command) to the keyboard driver input buffer. In many instances the prompt and command will be the same. An example of this is the word HELP, which is used by many applications.

We shall be looking more fully at the parameters and the vocabulary records as we work through the voice training program; so let's move on from the this introductory information and start on the training program itself.

As another example, to set up a word model for DIRECTORY on the second line, you would perform the command selection, and then type:

1. "2" for the line number followed by carriage return.
2. DIRECTORY for the PROMPT followed by carriage return.
3. "Y" for the COMMAND? field (to indicate that the command is not the same as the prompt).
4. DIR for COMMAND (the command to be sent to the computer) followed by carriage return and then FINISH.

You should now be able to see how the file of word-model records (prompts and commands) is built up.

Note: If you enter sufficient text to fill the box, the last character entered will also cause a carriage return to be actioned terminating any further text entry.

The DELETE command allows you to delete a record: just select DELETE and enter the line number followed by carriage return. This deletes the lines and re-orders the vocabulary file.

You should take great care when using this command especially within a file associated with a Voice Driven Application. The application expects to see word models in specific places. If you re-order the file using DELETE, the application may perform erroneous operations leading to catastrophic results.

On completion of editing a vocabulary file you have to go to the actual voice training routine.

Select FINISH to return you to the MAIN menu.

Training the Vocabulary

Before training a vocabulary you should first check that you have selected the correct file and that the parameters are set to the conditions in which the vocabulary will be used.

Now select the TRAIN WORDS option from the MAIN menu. The training menu now appears down the left-hand of the screen as shown below.

The command will not be excecuted because there is no carriage return. You are then be able to say either "WIDE" or "PAGE" which causes either "/W<CR>" or "/P<CR>" to be put into the computer input buffer, initiating the directory listing function.

Let's now look at the format required for adding or changing an entry on the screen.

Selecting ADD (new entry) or CHANGE (existing entry), then requires you to type in text in the following format.

1. The line number followed by carriage return (alternatively select the line number with the mouse and press the mouse button).
2. Type in the PROMPT followed by carriage return.
3. Type in the COMMAND? (Y or N).
4. Type in the COMMAND if Y was selected in COMMAND? field and press carriage return.

For example, to set up a word model for HELP on the first line, you would select ADD if it is a completely new file or CHANGE if it is an existing file and then type:

1. "1" for the line number followed by carriage return.
2. "HELP" for the prompt, followed by carriage return.
3. "N" for the COMMAND? field followed by a carriage return.
4. Select FINISH to signify the end of the command sequence.

Now if the word model for the prompt "HELP" is trained and recognised, the text HELP is sent to the computer everytime the word is said.

Using the Voice Training Program

This section on the VIS concentrates on the usage of the training program, Vtrain. It can be used for a number of different reasons, as follows:

1. Creating vocabulary files for either the Voice Function Key or the Voice Driven Application mode.
2. Editing/deleting existing vocabulary files.
3. Training/re-training a vocabulary.
4. Testing the accuracy of VIS to recognise words in a vocabulary.

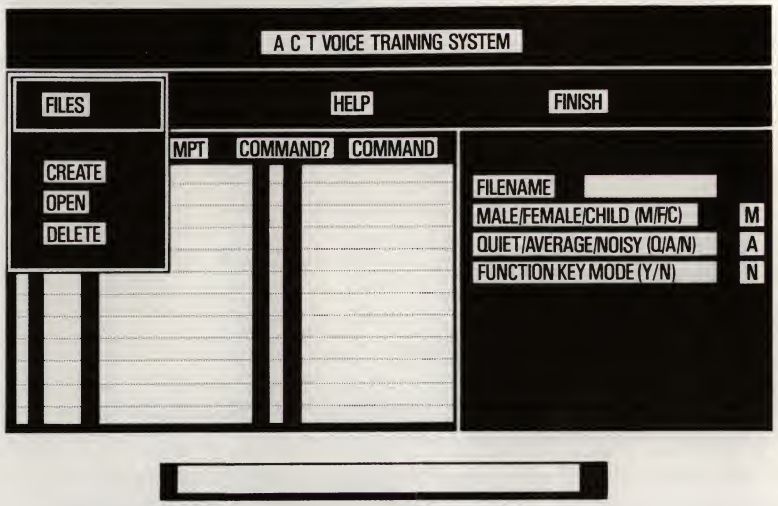
Loading the Training Program

The voice training program is started by simply loading the utility from the TRAIN.BAT batch file.

When used with the Apricot Portable system, optional parameters can be supplied with the file for switching the training program to run on either the LCD display or on an optional display monitor. The command options for the Apricot Portable are:

A>train	followed by <CR> to display the program on the default display (the one currently in use).
A>train pl	followed by <CR> to display the program on the LCD.
A>train pc	followed by <CR> to display the program on the monitor.

Type the command option from the operating system prompt and wait a few moments for the program to load and start. When it does, the screen is cleared and a menu is displayed, which looks similar to the one below.



This is a function selection menu (FILES) which is your entry point into the training program. It allows you the choice of specifying a vocabulary file and also allows you to select the operation you want to do on the vocabulary file.

It contains the following optional command selections displayed down the left-hand side:

CREATE
OPEN
DELETE

Special Characters

There are two characters which are interpreted by the VIS in a special way.

In some applications the use of a ∧ (caret) indicates the use of a control key. However we use ∧*M* to act as a switch for the carriage return function. For ease of use we normally assume that both the PROMPT and COMMAND fields are normally terminated by a carriage return.

If however a carriage return is *not* required to signify that the word used is not an actionable command, ∧*M* must be used at the end of the entry to indicate the suppression of the carriage return.

Another special character used is the underscore _, and simply denotes that a space character is to be sent to the computer input. You cannot use a space to separate different words, you must use an underscore instead. (e.g. *TYPE_^M* in the COMMAND field would allow you to speak the MS-DOS TYPE command and allow you to directly type in, the name of the filename + carriage return, before any action is taken).

Using these two special characters allows you to build up multiple word commands.

An example using the ∧*M* character to build a number of commands together before executing any of them is illustrated with the system MS-DOS command DIR. This command can be qualified by a suffix before MS-DOS processes it (a wide listing /W or a page listing /P). We can therefore build three word models to demonstrate this.

The first model contains "DIRECTORY" as the PROMPT, "Y" as the COMMAND? and "DIR ∧ M" as the COMMAND.

The second word model contains "WIDE" as the PROMPT, "Y" as the COMMAND? and "/W" as the COMMAND.

The third word model contains "PAGE" as the PROMPT, "Y" as the COMMAND? and "/P" as the COMMAND.

During training you will select the words DIRECTORY, WIDE and PAGE to be trained.

When the VIS is running within the operating system level, you are able to say "DIRECTORY", causing "DIR" to be put into the computer input buffer and displayed on the screen.

Selecting *REVIEW* lets you review another page of entries and is in effect a page up/page down command. If this command is selected you move from page to page either by using the mouse buttons (left for up/right for down) or ENTER/SHIFT + ENTER on the keyboard.

Before we show you how to edit a vocabulary file, we'll discuss the format and make-up of an entry in a vocabulary file. This was briefly touched upon in a previous section.

Select ADD to give you a clear view of all the columns on the Screen.

Each entry in the table is identified by the following fields:

1. A line number (WORD)
2. A PROMPT
3. A COMMAND? option
4. A COMMAND

WORD

The function of the line number WORD is self-explanatory, it justs specifies the line the entry occupies within the file.

PROMPT

This contains the word(s) which you will be prompted to say during training. Any text up to 16 characters can go in here. This text may also be the same as the command to be sent to the computer For example, HELP can be a prompt for a word, and also a command to the computer.

COMMAND?

This is a single character "Y" or "N" to indicate whether the text in the prompt field is different from the text that is to be sent to the computer on recognising this word.

"*N*" in this field indicates that the prompt is not different from the command to be sent to the computer input.

"*Y*" indicates that the command to be sent to the computer input is different from the prompt, and that it follows in the next field.

COMMAND

This field contains the command string to be sent to the computer input if COMMAND? is set to "*Y*". It may contain special codes such as control keys.

These are the initial entry level operations that you can perform on a vocabulary file.

CREATE allows you to create a new vocabulary file.

OPEN allows you to amend an existing vocabulary file. This could be adding or deleting words or re-training/ training a vocabulary.

DELETE allows you to delete an existing vocabulary file.

You'll also find two other commands across the top of the screen. These are HELP and FINISH. You'll find these two commands throughout all the menus within the training program.

Selecting HELP displays a message on the screen describing what functions and operations can be done in the selected menu.

Selecting FINISH returns you to either the menu you were in prior to the entry of your current menu, or out of the training program into the operating system environment.

The determining factor of where you exit to depends on where you are within the training program. If you selected FINISH now, you would exit out of the training program. As you step through the menus of the training program off the MAIN menu, you will find that selecting FINISH always returns you to the previous menu.

Before we illustrate how to perform specific operations with the training program, we'll show you the various methods available to for selecting a command. The same procedure can be used throughout all the menus within the training program.

Command Selection

You can make a command selection from a menu by one of three methods. These are as follows:

1. Typing in the correct text on the keyboard and pressing the carriage return key.
2. Placing the cursor over the required selection using the Mouse and pressing either one of the mouse buttons.
3. Using the keys on the numeric keypad (after holding down the SHIFT key) to move the cursor over the required selection and then pressing the ENTER key. In effect simulating a mouse. (Use the numeric keys 4 and 6 for left and right horizontal cursor movements; the numeric keys 8 and 2 for up and down cursor movements. Numeric keys 1, 3, 7, 9 can be used for diagonal movements. ENTER simulates the left mouse button, SHIFT + ENTER represents the right mouse button).

These three selection techniques can be used at all times within the voice training program. If you enter text once you are within a menu, the arrow cursor will normally disappear. To re-activate the cursor, you must first press the carriage return key.

There is also an easier way of selecting the command options, HELP and FINISH. These can be actioned by simply pressing the HELP and FINISH keys.

Starting

The first choice to be made before anything else can take place in a voice-training session is to select the operation you want to perform on a vocabulary file.

Select CREATE if you are going to produce a new file.

Select OPEN if you want to update or train an existing file.

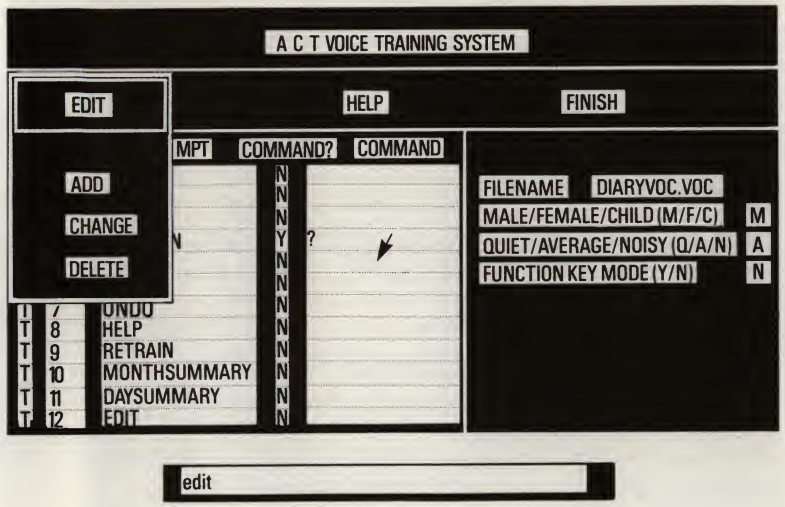
Select DELETE if you want to erase an existing vocabulary file.

Following your selection, the display now prompts you for the name of a vocabulary file, so type in the name filename required and press the carriage return key. (The .VOC suffix is not required).

If you make a mistake with your selection, simply repeat the operation. HELP displays a help text explaining the use of the parameters.

Editing Files

If you want to edit an existing file or begin the necessary process required for creating entries in a new vocabulary file, select EDIT. The main menu is now overlayed with the editing command options as illustrated below.



The command options available and functions of each command are as follows:

ADD allows you to add a new entry to the file.

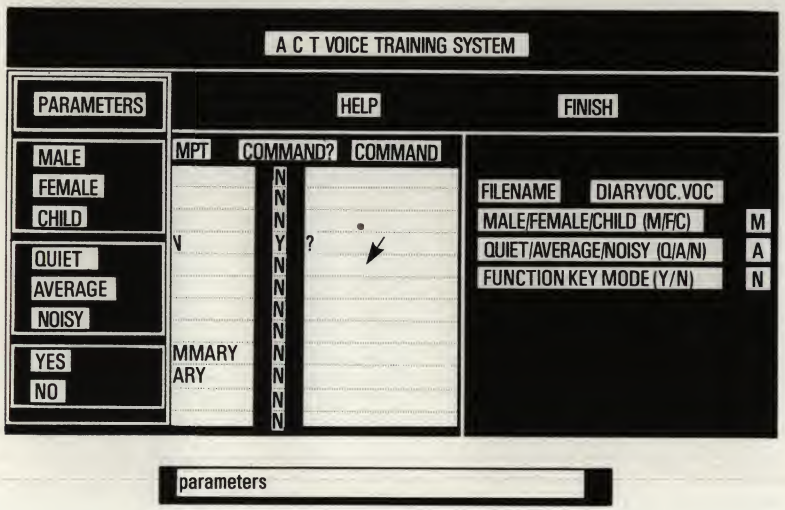
CHANGE allows you to change an existing entry within the file.

DELETE allows you to delete entries from the file.

Once one of the edit command options has been selected, one other command option is provided on a pop across menu. This is a review feature.

Changing the Parameters

The parameters can be edited at any time: select the PARAMETERS option to give the pop-down/pop-across menu as shown below.



Let's look at these parameters in a bit more detail:

MALE/FEMALE/CHILD

Indicates to the recognition program the pitch of your voice. You should choose the appropriate selection and then move to the next parameter.

QUIET/AVERAGE/NOISY

Indicates to the recognition program the level of background noise that is present in your working environment. Again, choose the appropriate parameter and then move to the next parameter.

FUNCTION KEY MODE

Indicates whether the vocabulary file is to be used in Voice Function Key mode (Y) or Voice Driven Application mode (N).

Select the parameters as required, and terminate parameter selection with FINISH to return you to the MAIN menu.

If using a voice driven application, the documentation supplied with the application will tell you the name of the file to load.

If you are configuring the system yourself in the Voice Function Key mode, you must first think of a suitable filename if it is your first time of use (e.g. for SuperCalc, use SCALC.VOC).

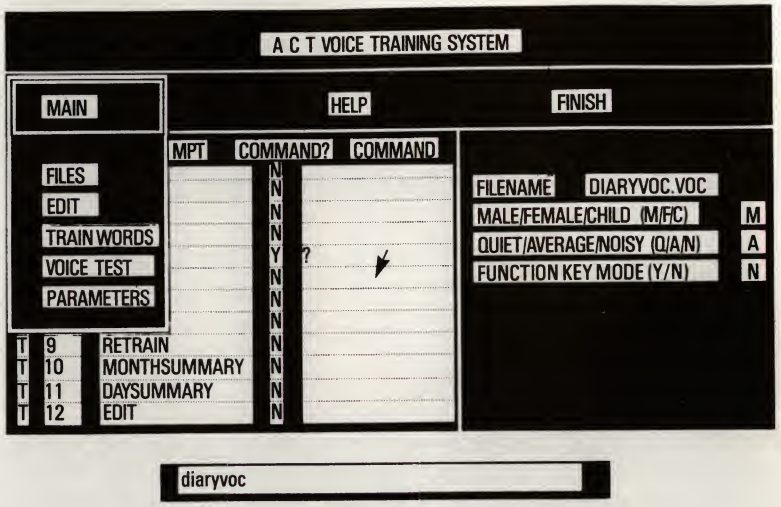
If you try to CREATE a file that already exists, or DELETE or OPEN a file that does not exist, an error message will be displayed.

Deleting a Vocabulary File

Selecting DELETE clears out all entries within the file, and erases the file completely. If the delete function is the only operation you wish to do, you can return to the operating system (or other program point of entry) by selecting FINISH.

Creating/Opening a File

Selecting either the CREATE or the OPEN command option changes the display to look like the one shown below (termed the MAIN menu). If you selected CREATE, the columns in the display will be blank since you are about to create a new vocabulary file. If the selection is OPEN, the columns will already contain some entries. A command menu overlays the left-hand side of the screen which specifies what operations you can do.



The main menu displays the filename of the file being created (or to be amended) and the current or default parameters.

This is the menu from which you can perform all operations on a vocabulary file. It allows you to edit entries, train the words in the vocabulary, change the operating parameters, and test the recognition capability of the VIS in its response to your voice.

You can view a summary of the function of each command option available from this menu by selecting the HELP command option, which will provide you with a condensed form of the information shown below and also prompts you to select a command.

EDIT allows you to change, add, insert or delete entries in the vocabulary file.

TRAIN WORDS allows you to train/re-train either individual words or all the words within a vocabulary.

PARAMETERS allows you to tailor the VIS to your status and working environment.

FILES returns you to the previous menu to allow you specify another vocabulary file to work on.

VOICE TEST allows you to test the recognisability of words in the vocabulary file.

Selecting FINISH terminates the training program and returns you to the operating system prompt.

If you only wish to train/re-train an existing vocabulary to respond to your voice and the parameter information for the file already corresponds to your status and work situation, you should now go directly to the section headed "Training a Vocabulary". If the operating parameters are incorrect, go to the section headed "Changing the Parameters" first, and then turn to the training section.

If you wish to edit an existing vocabulary file or create entries in a new file, check that the Parameters are correct by reading through the next section and then start on the editing section.

Application Interface

Apricot Voice Toolkit

Published in the U.K. by
ACT (UK) Ltd
Shenstone House
Dudley Road
Halesowen
West Midlands B63 3NT

Published in the USA by
Apricot Inc.
3375 Scott Boulevard
Suite 342
Santa Clara
California 95051

```
9005 VOICE$=""
9010 VOICE$=VOICE$+MKI$(COM%)
9015 VOICE$=VOICE$+FILE$+SPACE$(10-LEN(FILE$))          'vocab
9020 VOICE$=VOICE$+MKI$(OFF%)
9025 FOR I= 1 TO 8
9030      VOICE$=VOICE$+MKI$(MASK%(I))
9035 NEXT I
9040 VOICE$=VOICE$+MKI$(FLG%)
9045 VOICE$=VOICE$+TEXT$
9050 VOICE$=VOICE$+MKI$(CHK%)
9055 VOICE$=VOICE$+MKI$(CP%)
9060 VOICE$=VOICE$+HEAD$+SPACE$(36-LEN(HEAD$))
9065 VOICE$=VOICE$+MKI$(STAT%)+MKI$(REDO%)+MKI$(FILE%)+MKI$(ACTV%)
9070 CALL VI(VOICE$)

9075 REM ---------get status and other outputs----------
9085 STAT%=CVI(MID$(VOICE$,89,2))
9090 IF STAT% THEN PRINT "command : "COM% " "NOK$(STAT%) : END
9100 FILE%=CVI(MID$(VOICE$,93,2))
9105 ACTV%=CVI(MID$(VOICE$,95,2))
9115 RETURN

9900 DATA &H55                          : REM push bp
9901 DATA &H56                          : REM push si
9902 DATA &H8B,&HEC                     : REM mov bp,sp
9903 DATA &H8B,&H6E,&H08                : REM mov BP,8[bp]
9904 DATA &H8B,&H76,&H01                : REM mov si,1[bp]
9905 DATA &HCD,&HF6                     : REM int F6H
9906 DATA &H5E                          : REM pop si
9907 DATA &H5D                          : REM pop bp
9908 DATA &HCA,&H02,&H00                : REM ret 2

9910 VIFACE$=SPACE$(255)
9920 VI=PEEK(VARPTR(VIFACE$)+1)+(256*PEEK
     (VARPTR(VIFACE$)+2))
9950 FOR I=0 TO 16:READ J%: POKE VI+I,J%: NEXT I: RESTORE
9990 RETURN
```


Apricot Voice Toolkit: Application Interface

Contents

Introduction

This booklet forms part of the Apricot Voice Toolkit. It is a reference guide for Application writers.

The Apricot Voice Toolkit comprises the following booklets:

1. Overview
2. The TRAIN utility
3. Application interface

This part of the Voice Toolkit describes how an Application can interface to the Apricot VIS (Voice Input System).

There are two areas for Application writers to consider, they are:

1. How to link the Voice driver to the system
2. How to interface to the VIS from an Application.

For an overview of the system and details of how to set up and train vocabulary files refer to the other booklets mentioned above.

The remainder of this booklet assumes that the reader is familiar with the techniques of vocabulary creation and training.

```
525 IF CALC$="select" THEN 320
530 IF CALC$="end" THEN PRINT "End of program":END
535 IF CALC$="help" THEN GOSUB 6000: GOTO 505
540 I=1
545 TEMP$=""
550     WHILE MID$(CALC$,I,1) > "/" AND MID$(CALC$,I,1) < ":"
555         TEMP$=TEMP$+MID$(CALC$,I,1)
560         I=I+1
565   WEND
570 NUM1%=VAL(TEMP$) : TEMP$=""
575 OP$=MID$(CALC$,I,1): I=I+1
580   WHILE I <= LEN(CALC$)
585         TEMP$=TEMP$+MID$(CALC$,I,1)
590         I=I+1
595   WEND
600 NUM2%=VAL(TEMP$)
605 MESS$="result is : ":ON INSTR("+-/*",OP$)+1
    GOSUB 1000,2000,3000,4000,5000
610 PRINT MESS$ ; RES
615 GOTO 505

1000 MESS$="invalid separator":RES=0:RETURN
2000 RES=NUM1%+NUM2%:RETURN
3000 RES=NUM1%-NUM2%:RETURN
4000 RES=NUM1%/NUM2%:RETURN
5000 RES=NUM1%*NUM2%:RETURN

6000 PRINT CLS$"You may say the following commands:"
6010 PRINT: PRINT   "'END' to end the program"
6020 PRINT          "'PICK-LANGUAGE' to choose another language"
6030 PRINT          "an expression of 2 integers including one operator:"
6040 PRINT"           for + say 'PLUS"
6050 PRINT"           for - say 'MINUS'or 'MOINS'"
6060 PRINT"           for * say 'MULTIPLY BY' or 'MULTIPLIER PAR'"
6070 PRINT"           for / say 'DIVIDED BY' or 'DIVISEZ'"
6075 PRINT"           for = say 'IS EQUAL TO' or 'EGAL'"
6080 PRINT" 'BACK' to backspace"
6090 RETURN

7000 PRINT CLS$"This is a simple integer calculator which allows you to enter"
7010 PRINT"an expression in either English or French"
7030 PRINT"Levels 1 & 2 of the Speech Driver are active to provide the"
7040 PRINT"following commands:"
7050 PRINT
7060 PRINT"'END' (level 1) to terminate the program"
7070 PRINT"'HELP' (level 1)"
7080 PRINT"'ENGLISH' or 'FRENCH' (level 2)"
7090 PRINT
7100 PRINT"When a language is chosen then it's vocabulary is loaded (level 3)."
7110 PRINT"The other language remains inactive.
7120 PRINT"The vocabulary at level 2 is made inactive but level 1 remains active"
7130 COM%=4:FLG%=1:GOSUB 9000
7140 INPUT"say 'END' to continue"; OK$
7150 COM%=4:FLG%=0:GOSUB 9000
7190 RETURN

9000 REM ------build voice interface string and call routine------
9001 GARBAGE=FRE(""):GOSUB 9900
```

```
10 CLS$=CHR$(27)+"E":PRINT CLS$;
30 NOK$(1)="init. error"
40 NOK$(2)="no vocabulary"
50 NOK$(3)="too many words"

100 REM----------initialise voice and check vocabulary----------
105 CHK$="end"+SPACE$(13)+"help"+SPACE$(12)   '1st 2 entries
110 COM%=1                                    'command
115 FILE$="MULTI"                             'vocabulary file
120 OFF%=0                                                        'mask offset
125 FOR I= 1 TO 8: MASK%(I)=0 : NEXT I        'mask
130 FLG%=2                                    'voice under program control
135 TEXT$=SPACE$(16)                          '
140 CHK%=2                                    'chk 2 entries
145 GARBAGE=FRE(""):CP=PEEK(VARPTR(CHK$)+1)+256*PEEK(VARPTR(CHK$)+2)
150 IF CP >32767 THEN CP%=CP-65536! ELSE CP%=CP
155 GOSUB 9000

300 REM-----------set level 1 recognition-----------
305 COM%=5:FILE$="MULTI":OFF%=0:MASK%(1)=&H3:MASK%(2)=&H4000:
    MASK%(3)=0
310 GOSUB 9000                                'do command
315 PRINT CLS$: INPUT "Please press the shift+VOICE keys"; A

319 REM-----------set level 2 & reset level 3-----------
320 PRINT CLS$"Say 'HELP' or select a language : ";
321 COM%=6:FILE$="MULTI":MASK%(1)=&HC:MASK%(2)=0:MASK%(3)=0:
    GOSUB 9000
322 COM%=7:FILE$="MULTI":MASK%(1)=0:MASK%(2)=0:MASK%(3)=0:GOSUB 9000

325 COM%=4:FLG%=1:GOSUB 9000
330 INPUT LANG$
335 COM%=4:FLG%=0:GOSUB 9000

336 COM%=6:FILE$="MULTI":MASK%(1)=0:MASK%(2)=0:MASK%(3)=0:GOSUB 9000

340 I=INSTR("english/french/help/end",LANG$):IF I=0 THEN 320
350 IF LANG$="end" THEN PRINT "end of program":END
360 IF LANG$="help" THEN GOSUB 7000:GOTO 320
400 REM-----------set level 3 recognition-----------
410 COM%=7:FILE$="MULTI":OFF%=0
420 IF LANG$="english" THEN MASK%(1)=&HFFE0:MASK%(2)=&H800F:MASK%(3)=0
425 IF LAG$="french" THEN MASK%(1)=&H200:MASK%(2)=&HBFF0:MASK%(3)=&H1F
430 GOSUB 9000

490 REM--------------------------------------
500 REM - main loop
505 PRINT "Please say 'help' or say an expression ";

510 COM%=4: FLG%=1: GOSUB 9000
515 INPUT CALC$
520 COM%=4: FLG%=0: GOSUB 9000
```


Voice Driven Applications

Applications configured to run with Voice are simply enhanced versions of those which use the Keyboard. The functional difference being that when the Voice software is active, voice input is matched against word models and then translated into data prior to being placed into the Keyboard queue.

This means that both the Keyboard and the Voice input device may be used simultaneously.

The Speech driver cannot become active without intervention initially by the User, i.e. depression of the VOICE key (SHIFT + F5). This ensures that spurious noise does not cause input to the Application until it is ready.

This action is not obvious to the untrained user.
For this reason it is suggested that all Voice Driven Applications should prompt the User to press the VOICE key when input is expected following loading of the application.

Thereafter, the application itself may switch the Speech driver on and off as required thus ensuring that the operation of the program is not affected by random noise input.

The Speech driver only recognises a particular set of vocabulary word models at any one point in time as specified by the Application.

With good training and a reasonable environment the performance of the driver is very accurate. However, it is not infallible!

Random noise or incorrect commands which have a similar pattern to an active word in the vocabulary will result in an unwanted match from time to time.

In such cases, the Speech driver will send the matched command to the Keyboard queue. The Application should take steps to validate the input just as it would if incorrect commands were typed in.

The Speech driver can always be deactivated by the User by pressing the VOICE key (SHIFT + F5). The Application will then work just as a normal Keyboard input-based Application until the User reactivates it; once again by pressing the VOICE key.

Features of VIS

The VIS provides the Application with a simple set of commands which support the following features:

1. Switching the Speech driver on and off.
2. Vocabulary selection
3. Formation of a vocabulary hierarchy
4. Vocabulary checking

There are 5 commands which enable the Application writer to control and set up the Speech driver. Each one requires a 48 word parameter block of exactly the same structure.

A vocabulary of up to 63 words may be designated active at any one time but the total number of usable words is limited only by disk size and practicality.

Any number of vocabulary files may be created using the TRAIN utility. Each file is limited to 9999 words but the VIS allows selections to be made from up to a maximum of 30 vocabulary files in any one Application.

The VIS commands also enable the Application to create a 3 level hierarchy of vocabulary.

Groups of vocabulary records may be masked in and out of each level as the context of the Application changes, while others may remain active throughout. This enables the Speech driver to disregard spurious noise and incorrect commands to a reasonable degree.

The validation options simply ensure that the vocabulary files to be used by an Application are correct. The validation takes place either on the entry in the command field in the vocabulary file if it differs from the prompt, or if the prompt and the command are the same, the prompt is used for validation.

Similarly, the use of the string to validate vocabulary results in the same problem and the solution is illustrated in statement 145.

Secondly, in Basic the variable pointers in the case of strings do not point to actual strings but instead to a 3 byte packet consisting of:

String byte count
Address of string data (2 bytes)

The machine code interface is used to pick up the address of the parameter block and in addition set's up DS:SI as required in a similar way to the other examples.

In the example we are also using another string to point to the vocabulary check data. Statements 145 and 150 show how to pick up the address of the actual string in Basic as opposed to machine code.

Apart from these points the program should read just as any other Keyboard Application with the addition of the requirements of setting up the parameter block.

For this purpose a brief overview of some of the key statements is given here to assist comprehension:

100 - 155	initialise voice and check the first 2 entries in the vocabulary.
305	selects the level 1 vocabulary which remains active throughout the Application.
321	selects the level 2 vocabulary so that the choice of language can be given, i.e. either 'english' or 'french'.
322	deactivates any selections already made at level 3.
336	deactivates level 2.
410 - 425	activates level 3 dependant upon the selection made at level 2.

The entries are used in the example as follows:

Level 1 Always active 1 and 2

Level 2 Only active when choosing the language:

3 english
4 french
31 back i.e. backspace

Note: German is not activated by the program.

Level 3 The following sets are activated:

English - entries 6 thru 20 and 32
(pick_language)

French - entries 21 thru 30 and 32 thru 37.

Note: The use of different prompts to produce the same command is used effectively in the file given above. However, there are some similarities which may cause problems especially if the training conditions are not perfect.

The example program in interpretive Basic below serves to illustrate the use of multiple levels within an Application.

An explanation of some of the peculiarities of Basic and the solutions employed to solve them may help before reading it.

Firstly, as in all the other examples an interface routine in machine code is required to set up the DS:SI register and invoke the interrupt F6 hex. This is provided by the subroutine, starting at statement 9900,which creates a machine code interface in the string variable VIFACE$.

The parameter block is created by successive concatenations of the data fields in the subroutine starting at statement 9000. String concatenation, however, results in Basic using large ammounts of RAM and without any direct intervention the program will run out of memory.

For this reason a garbage collection is carried out regularly at statement 9001. This results in all strings being reorganised and the possibility of a change in location in RAM which may effect the start address of the machine code interface. Therefore after every garbage collection the interface is rebuilt.

Neater solutions can be derived by using integer arrays to hold the routine but the program deliberately incorporates this solution to emphasise the problems of interfacing.

Installing the Voice Driver

The Speech module is a loadable device driver which is released as a file called SPEECH.SYS.

It is loaded at BOOT time provided that the file is present on the BOOT disk and that a corresponding entry is found in the CONFIG.SYS file. To check the entries simply type:

```
A>type config.sys
buffers=10
country=44
device=mouse.sys
device=speech.sys
```

and typically the above information will be displayed on a release disk.

When the driver is loaded a release message is displayed together with Speech driver copyright details.

However, before voice can be used the ACT voice interface must be loaded. This is normally achieved by invoking a batch file which is supplied on the release disk. The required command is:

```
A>voice <filename> <word models>
```

The command requires two parameters:

1. To specify the name of the vocabulary file to be used by the application
2. To specify the buffer space to be reserved for the resident word models.

The <filename> should refer to a valid MS-DOS file which has been created by the TRAIN utility. It must have an extension of VOC. Normally the <filename> loaded is the first required by the Application although any valid vocabulary file may be invoked here.

The <word models> parameter reserves buffer space for the vocabulary records to be used in the Application.

The Speech driver has reserved space for up to 63 word models. These are the word models which are resident as opposed to active. If the Application uses more than 63 word models then in order to increase the speed of response it is advisable to allocate further memory for as many of them as possible. Failure to do this will result in the slowing down of the Application due to disk accesses.

The VIS uses the buffer as a cache. All word models selected to be active are read from the disk and placed in the buffer. If there is insufficient space in the buffer to introduce one or more new word models to the Application, then non-active models are released from the cache to make room for the new ones.

The number specified in the parameter <word models> must include the minimum reserved space of 63.

For example, if an Application uses up to 100 word models throughout execution, even though a maximum of only 20 are ever active at any one point in time, then ideally it should still specify <word models> = 100. In any case it is recommended that a minimum figure of 63 is always reserved.

The Speech driver will require approximately 300 bytes of RAM for each word model above 63, i.e. for the case outlined above (100 - 63) x 300 which is approximately 11,000 bytes.

A different vocabulary file called MULTI.VOC is used in this case. It should contain the following word models:

Entry	Prompt	Command?	Command
1	end__program	Y	end
2	help	N	
3	english	N	
4	french	N	
5	german	N	
6	plus	Y	+∧M
7	minus	Y	—∧M
8	multiply-by	Y	*∧M
9	divided__by	Y	/∧M
10	is__equal__to	Y	__=
11	one	Y	1∧M
12	two	Y	2∧M
13	three	Y	3∧M
14	four	Y	4∧M
15	five	Y	5∧M
16	six	Y	6∧M
17	seven	Y	7∧M
18	eight	Y	8∧M
19	nine	Y	9∧M
20	zero	Y	0∧M
21	un	Y	1∧M
22	deux	Y	2∧M
23	trois	Y	3∧M
24	quatre	Y	4∧M
25	cinq	Y	5∧M
26	six	Y	6∧M
27	sept	Y	7∧M
28	huit	Y	8∧M
29	neuf	Y	9∧M
30	zero	Y	0∧M
31	back	Y	∧H∧M
32	pick__language	Y	select
33	multiplier__par	Y	*∧M
34	divisez	Y	/∧M
35	plus	Y	+∧M
36	moins	Y	—∧M
37	egal	Y	__=

Example 3: Microsoft Interpretive Basic

The Calculator program given in the previous examples is expanded in this example to show how the levels are used effectively within an Application.

The program is fundamentally the same except that a second level choice is given to select a language from English or French. Then the third level is introduced as the actual vocabulary for entering an expression in the language chosen.

As the program uses several levels it also illustrates how sets of vocabulary may be activated/deactivated according to the context of the Application.

Voice Interface

Overview

An Application interfaces with the Speech driver via the Voice module of VIS which must be pre-loaded as described above.

A software interrupt (interrupt F6 hex) is reserved for the Application to link to the VIS.

The Application controls the Speech driver by generating various commands. This involves constructing a parameter block and then invoking the F6 interrupt with a Segment:Offset pointer in the DS:SI registers.

The Commands provide the following facilities:

1. Loading from multiple vocabulary files.
2. Validation checks to ensure the integrity of the vocabulary file.
3. Selection of word models from the vocabulary files.
4. Specification of a 3 level hierarchy. Further details are given in the sub section 'Word model selection'.
5. Ability to switch Voice recognition on/off. This ensures that spurious noise does not result in data affecting the normal course of action of the Application.

The VIS returns data to the Application in the same parameter block.

Certain commands are reserved for future implementation. The active commands are summarised in the following section.

Command summary

A brief description of each command is tabled below:

Command	Description
01	Load a word model file and optionally: 1. Check specific entries within the file to ensure integrity. 2. Select an initial vocabulary set from the file.
02	Reserved for future implementations.
03	Reserved for future implementations.
04	Activate/Deactivate Speech driver. This allows the Application to turn the Speech driver on and off to coincide with input requests. This avoids the detection of spurious noise which may generate random inputs at undesirable points in the Application.
05	Select vocabulary - Level 1 This is the highest level of the vocabulary hierarchy. The vocabulary set selected at this level remains active when selections at the two lower levels are made.
06	Select vocabulary - Level 2 Selections may be made at Level 2 to complement those at Level 1. Applications may invoke different sets of vocabulary at this level thus forming the hierarchy. The vocabulary set selected at this level remains active when selections are made at the higher and lower levels.
07	Select vocabulary - Level 3 Selections at this level complement those at the two higher levels. Different sets of vocabulary may be invoked according to Application needs.
08	Kill voice - not currently implemented

```
3000    if (c$ = "+" or c$ = "-" or c$ = "*" or c$ = "/") then goto 3010
        sflag = 1
        return
'
3010    sflag = 0
        return
'
' this subroutine checks the returned status byte and if it is not zero will
' display a message and then halt execution of the program.
'
4000    status = cvi(mid$(temp$,90,1) + mid$(temp$,89,1))
        if (status = 0) then return
        if (status <> 1) then goto 4010
        print
        print "*** ERROR *** in command ";command
        print "returned code is 01"
        print
        stop
'
4010    if (status <> 2) then goto 4020
        print
        print "*** ERROR *** in command ";command
        print "returned code is 02"
        print
        stop
'
4020    print
        print "*** ERROR *** in command ";command
        print "returned code is 03"
        print
        stop
'
' This subroutine builds up the parameter sting to be passed to the voice
' system.
'
5000    temp$ = ""
        temp$ = temp$ + mki$(command)
        temp$ = temp$ + filename$
        temp$ = temp$ + mki$(offset)
        for i = 1 to 16
        temp$ = temp$ + left$(mki$(mask(i)),1)
        next i
        temp$ = temp$ + mki$(iflag)
        temp$ = temp$ + text$
        temp$ = temp$ + mki$(instance)
        temp$ = temp$ + mki$(vptr)
        temp$ = temp$ + otext$
        temp$ = temp$ + mki$(status)
        temp$ = temp$ + mki$(indicator)
        temp$ = temp$ + mki$(nowms)
        temp$ = temp$ + mki$(nowmsact)
        call filter(temp$)                    'call the interface routine
        return
```

```
            ' switch voice system off
            ' ----------
            '
            ' this is a command "04". The input flag signifies that the voice
            ' system should be switched off.
            '
            iflag = 0
            gosub 5000
            return
            '
' This subroutine takes the input line and checks the first character to see
' if it is either "?" - help, or "E" - finish. If it is not, it will then
' separate out the first number, the operand and the second number.
'
2000        '
            ' test for "E"
            '
            operand$ = left$(line$,1)
            if (operand$ = "E") then return
            '
            ' test for "?"
            '
            if (operand$ = "?") then return
            '
            ' separate out the first number
            '
            temp$ = ""
            for i = 1 to len(line$)
            c$ = mid$(line$,i,1)
            gosub 3000                      'is character an operand?
            if (sflag = 0) then goto 2010   'yes, so convert string to number
            temp$ = temp$ + c$              'no, so just add character to string
            next i
2010        number1 = val(temp$)
            '
            ' separate out the operand
            '
            operand$ = mid$(line$,i,1)
            '
            ' separate out the second number
            '
            j = len(line$) - i
            temp$ = right$(line$,j)
            number2 = val(temp$)
            return
'
' this subroutine checks the supplied character to see if it is a seperator,
' ie + , — , / or *. If it is, then 0 is returned, if not, then 1 is
' returned.
'
```


The Parameter Block

The Voice interface requires a 48 word parameter block. The block has an identical structure for each command but only certain fields are relevant in each case. As stated above commands 02 and 03 are reserved for future use as is command 08. In addition certain fields within the parameter block are also reserved. A summary of the fields and their use by each command is tabled below:

No.	Field desc.	Size (words)	Start addr.	Field id					Data type
				01	04	05	06	07	
Inputs:									
1	Command	1	1	0	1	0	4	05 06 07	Integer
2	Filename	5	2	Y	x	Y	Y	Y	Ascii
3	Offset	1	7	Y	x	Y	Y	Y	Integer
4	Mask	8	8	Y	x	Y	Y	Y	" "
5	Flag	1	16	Y	Y	x	x	x	" "
6	Reserved	8	17						
7	Check	1	25	o	x	x	x	x	Integer
8	Pointer	1	26	o	x	x	x	x	" "
Outputs:									
9	Reserved	18	27						
10	Status	1	45	Y	Y	Y	Y	Y	Integer
11	Reserved	1	46						
12	Filesize	1	47	Y	x	Y	Y	Y	Integer
13	Active	1	48	Y	x	Y	Y	Y	" "

The Field id column indicates which fields are required by each command. The key is:

Y - Yes, the input fields must be valid as specified below. The output fields are affected by the command.

x - Don't care or not affected.

o - Optional, the field may be optionally specified.

For example, command 05 must receive valid input data in fields 2, 3 and 4 (i.e. filename, offset and mask respectively). The output fields affected by the command are the Status field 10, the Filesize field 12 and the active field 13.

The format and a description of each field follows.

In general each field has the same format and significance to each command which uses it. Details of specific input and output settings are given in the detailed description of each command later.

The general format is:

Field	Description
Command	Either 01, 04, 05, 06 or 07
Filename	An MSDOS drive:filename specifier, i.e. d:ffffffff The vocabulary file suffix is assumed to be VOC and should not be included in the field. The drive specification is optional. A different filename specification may be used in each of the relevant commands. This provides the Application with a Multi file capability.
Offset	The offset specifies a block of 128 vocabulary records within the file. The Mask field below makes selections from the block specified here. An offset of 0 specifies the 1st block, 1 the 2nd and so on.
Mask	An array of eight 16 bit words used to select vocabulary records from the block specified in the offset above. Each bit has the following significance: 0 = deselect word model 1 = select word model Up to 63 words may be selected from a block of the vocabulary in any one command. However the total of those previously selected together with those currently requested in the mask must not exceed 63. The format and use of the mask is discussed later in the sub section 'Word model selection'

```
        if (operand$ <> "+") then goto 110
        result = number1 + number2
        goto 160
'
110     if (operand$ <> "-") then goto 120
        result = number1 - number2
        goto 160
'
120     if (operand$ <> "/") then goto 130
        result = number1 / number2
        goto 160
'
130     if (operand$ <> "*") then goto 140
        result = number1 * number2
        goto 160
'
140     if (operand$ <> "E") then goto 150
        end
'
150     print
        print "Help Message"
        print
        goto 100
'
160     print "Result is ";result
        print
        goto 100
'
' This subroutine outputs a prompt then switches the voice system on and
' waits for input. Once a line has been entered, either by voice or by
' keyboard, (the program can not tell the difference), the voice system
' is switched off and control returns to the calling program
'
1000    '
        ' output prompt
        '
        print
        print "Please enter your calculation";
        '
        ' switch voice system on
        ' ----------
        '
        ' this is a command "04". The input flag signifies that the voice
        ' system should be switched on.
        '
        command = 4
        iflag = 1
        gosub 5000
        '
        ' get input line
        '
        input line$
        '
```

```
command = 5
filename$ = vocname$
offset = 0
mask(1) = 7
for i=2 to 16
mask(i) = 0
next i
gosub 5000
'
' test the returned status byte
'
gosub 4000
'
' set up mask to enable level 2 recognition
' ------------------
'
' this is the command "06". Again the vocabulary file is specified
' and the offset means we wish to only consider the first 63 words
' in the file. The mask shows that we wish to activate the remaining
' words at this level. Again the returned status is checked.
'
command = 6
filename$ = vocname$
offset = 0
mask(1) = 248
mask(2) = 255
mask(3) = 3
for i = 4 to 16
mask(i) = 0
next i
gosub 5000
'
' test the returned status byte '
gosub 4000
'
' main loop of the program
' ------------
'
'
' read in a line, check for "?" or "E" and then perform the required
' calculation
'
100 '
' read in a line
'
gosub 1000
'
' check for "?" or "E" else break the line down into first
' number, operand and second number
'
gosub 2000
'
' perform the calculation
'
```


Field	Description
Flag	This field is command dependant and in general is used to initialise, activate and deactivate the Voice input device.
Check	The number of vocabulary words to be validated when the file is loaded. This is an optional field. If it is set to zero then no validation takes place and the next field is ignored.
Pointer	This is a pointer to a string of vocabulary commands which are to be validated. Each entry in the string must correspond with the command, or the prompt where it is the same as the command, which is entered into the vocabulary file in the TRAIN utility. The string must contain a 16 character field for each of the word models to be validated as specified in the Check field above. Each string should be left justified and space filled. The examples included later in the booklet serve to illustrate this.
Status	The status word is affected by all commands; in general the following values are returned: 00 Ok 01 Fail 02 Has three possible meanings depending upon the command used. They are: 1. File not found. 2. File empty. 3. Insufficient space for another file. Only 30 files may be accessed by an Application.
Filesize	Returns the number of available word records within the filename specified above.
Active	Returns the current total of word models which are active following the last command.

Word model selection

The Mask and Offset fields are used in commands 01, 05, 06 and 07. They enable the Application to select and deselect word models.

The use of Mask and Offset fields is the same for each command and therefore an overview of them is given below to simplify the description of each command.

A vocabulary file may theoretically contain up to 9999 word models. At any one time word models from one or more vocabulary files may be designated active.

A vocabulary file is considered to be in blocks of 128 word models and each block is designated by a value in the offset field.

Each selection is made by setting a bit in one of the eight elements of the 128 bit Mask array, the bit settings correspond with:

0 to deselect a word model
1 to select a word model

The first element maps the first 16 word models within the block, the second element maps the 17th to the 32nd and so on.

Each element maps the models in reverse order.
The figures below show the first two elements which correspond to the first 32 word models in the file.

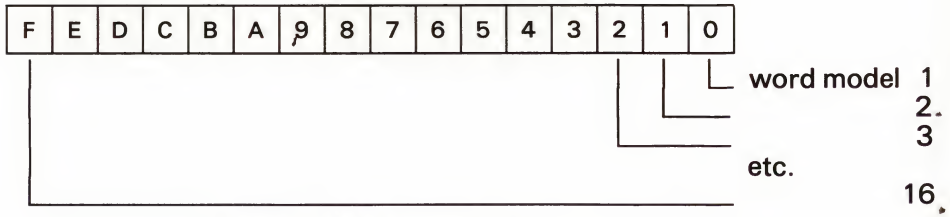


Figure 1: Mask element 1

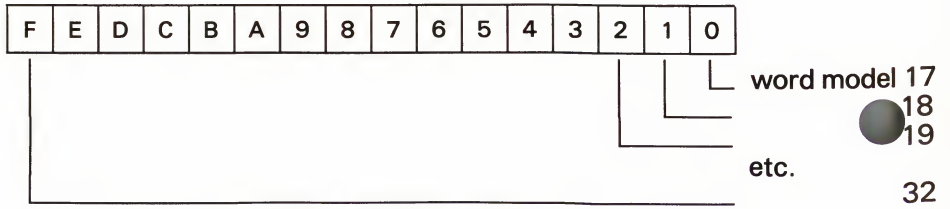


Figure 2: Mask element 2

Microsoft Compiled Basic

This is a sample program written in Microsoft Basic to show an application may use the ACT Voice system. This particular application is a simple integer calculator. N.B. This program should be compiled with the N switch.

```
defint a-z

dim mask(16)

text$ = space$(16)
otext$ = space$(36)
vocname$ = "CALCULAT"      ' name of vocabulary file to be used '
' The following line defines the first two entries in the vocabulary
' file. This will be used by the voice system to check that it has
' the correct file. N.B. The validation is made with the command
' strings, not the prompt string.
'
vcmd$ = "? ^ M              E ^ M"
'
' Initialise the voice driver
' ----------
'
' This is the "01" command. It defines the vocabulary file to be used
' and sets the offset within the file and the mask to zero. The input
' flag is set to signify that voice control is ON. The number of
' strings to be validated is set to 2 and that they are held in vcmds.
' A check is made on the returned status to confirm that all is OK
'
command = 1
filename$ = vocname$
offset = 0
for i=1 to 16
mask(i) = 0
next i
iflag = 2
instance = 2
vptr = varptr(vcmd$)
gosub 5000
'
' test the returned status byte
'
gosub 4000
'
' set up mask to enable level 1 recognition
' ----------
'
' this is the command "05". Again the vocabulary file is specified
' and the offset means we wish to only consider the first 63 words
' in the file. The mask shows that we wish to activate the first
' three words only at this level. Again the returned status is
' checked.
'
```

Interface for Microsoft Compiled Basic

```
        PAGE    60,132
        TITLE   Voice Interface Program
;
;An interface program for applications written in
;languages that pass pointers to data structures
;as a single word offset on the system stack.

CODE    SEGMENT BYTE    PUBLIC  'CODE'
        ASSUME  CS:CODE

        PUBLIC  VOICE
VOICE   PROC    NEAR

;This routine will be called with the following
;stack contents.

;                   STACK: param. block offset : +2
;                  (SP) -> : return address    :  0

        PUSH    BP              ;save working registers
        PUSH    SI
        MOV     BP,SP           ;point to stack.(param.block offset
                                ;will now be 4 bytes down the stack)
        MOV     SI,6[BP]        ;DS:SI will now point to the parameter block
        INT     0F6h            ;call the VOICE programs

        POP     SI              ;restore the working registers
        POP     BP
        RET     2

VOICE   ENDP
CODE    ENDS
        END
```


When vocabulary records are created, their order in the file need bear no relationship to the sequence required by an Application as it executes it's various tasks.

However, as can be seen from the bit settings of the mask, contention occurs within a block when an Application attempts to make successive selections from the same block. All bits marked 0 will deselect a word if it has already been selected. For example, if the Application takes the following steps:

1. Activate word models 1, 2 and 16. Then element (1) will have a setting of 8003H.
2. Activate word models 5 thru 15. Then element (1) will have a setting of 0FFEH.

Then the result will be that word models 1, 2 and 16 will be deactivated by the second call as the bits relating to these word models are zero.

In complex Applications it is impractical and inefficient to remember previous activations in order to reselect them.

Therefore, the VIS provides a number of calls which enable the Application to stucture a hierarchy of vocabulary.

These commands, as listed above, allow for multiple selections at 3 levels within a block, or in a number of blocks, without affecting previous selections at a different level. The following rules also apply:

1. If a selection of a word model is made and it is already active at the same or a higher level then it is ignored. No fail status is given and the word remains active.
2. If a selection of a word model is made and it is already active at a lower level then it is removed from the lower level and placed in the level requested.
3. With each command the Application can make selections at it's level provided that the total number of word models active in all levels does not exceed 63.
4. Within any one command a selection can only be made from a single block of 128 word models.

Applications which require word models from 2 or more blocks at any one level must make multiple calls to the same command and modify the offset accordingly, i.e.

Offset	Word Model range
0	1 - 128
1	129 - 256
2	257 - 384
etc.	

Programming examples in the final section of this booklet serve to illustrate the use of hierarchies, masks and offset settings.

Filter for Microsoft Compiled Basic

This is an interface routine required by Microsoft Compiled Basic because the basic compiler puts a pointer to a string descriptor onto the stack rather than a pointer to the string itself. The purpose of the routine is to pull the string descriptor pointer off the stack and to replace it with a pointer to the actual string.

```
code    segment byte    public  'code'
        assume  cs:code
        public  filter          ;make this routine available to basic
        extrn   voice:near      ;tell the assembler about voice
filter  proc    far             ;routines called by basic must be 'far' routines
        push    bp              ;save current base page register
        mov     bp,sp           ;copy the stack pointer
        mov     bp,6[bp]        ;load the address of the string descriptor
        mov     bp,1[bp]        ;get the address of the actual string
        push    bp              ;put it onto the stack for voice to find
        call    voice           ;call voice
        pop     bp              ;reset the base page register
        ret     2               ;return to basic
filter  endp
code    ends
        end
```

Example 2: Microsoft Compiled Basic

The program given below is exactly the same as Example 1 with the exception that an additional interface (or filter) is required to set up the pointer to the parameter block.

The file called CALCULAT.VOC is also used in this example.

The first three entries are designated by the Application as level 1. The remainder are designated level 2.

The two machine code interfaces must be linked to the program before the program can be used. Their purpose is to extract pointer information from the stack and set up the internal registers prior to calling the VIS via interrupt F6 hex.

A listing of each interface is given on the following pages.

Command specification

Load a word model file (command 01)

This is the first command which must be executed in any Voice Application. It's main function is to initialise the VIS and to specify the preliminary state of the Voice device.

The Flag field should be set to the following values:

Setting	Input	Application
0	off	off
1	on	off
2	off	on
3	on	on

The normal setting for an Application which is to control the Input device would be Flag = 2, i.e. Input device off and under Application control.

In addition the following functions may be performed:

1. Make a preliminary selection of vocabulary at the lowest level of the hierarchy. If the Mask array is set to all zeroes then no selections are made.

   Note that only the word models which are validated can be made active by this command.

2. Validate the vocabulary file chosen. Normally this will not be required as the vocabulary is unlikely to change in a released Application. However it is an extra safeguard. The command may validate any of the first 63 word models in the file. All word models from 1 to n, where n is the number to check, must appear in the validation string.

The command returns the Status word as defined above. If the operation was successful then the other return data will be valid and will give the following information:

Filesize - the number of word models in the file

Active - the number of word models, if any, activated by the command.

Activate deactivate Input Device (Command 04)

This command should be used when the Application is expecting input and after it has received input.

It requires a single parameter in the Flag field, which can have the following settings:

0 - turn Input Device off

1 - turn Input Device on

The Status byte may be returned with a value of 0 or 1 as defined earlier.

This command will not activate the Voice input device initially until the User has invoked voice by pressing the SHIFT +VOICE key.

```
{
  if (c IS '+' OR c IS '-' OR c IS '*' OR c IS '/')
  {
    return(FALSE);
  }
  else
1{
    return(TRUE);
  }
}

/* this subroutine checks the returned status byte and if it is not zero will
   display a message and then halt execution of the program.
*/

show__status()

{
  if (viface.status ISNOT 0)
  {
    if (viface.status IS 1)
    {
      printf("/n*** ERROR ***/nreturned code is 01/n");
      exit(1);
    }
    else
    {
      if (viface.status IS 2)
      {
        printf("/n*** ERROR ***/nreturned code is 02/n");
        exit(1);
      }
      else
      {
        printf("/n*** ERROR ***/nreturned code is 03/n");
        exit(1);
      }
    }
  }
}
```

```
if (line[0] IS FINISH)                    /* test for 'E' */
{
  operand = FINISH;
}
else
{
                                          if (line[0] IS HELP)          /* test for '?' */
  {
    operand = HELP;
  }
  else
  {
                                          /* separate out the first number */
    i = 0;
    while (not_separator(line[i]))
    {
      temp[i] = line[i];
      i++;
    }
    temp[i] = TERMINATOR;
    stcd_i(temp,&number1);
                                          /* separate out the operand */
    operand = line[i];
                                          /* separate out the second number */
    i++;
    j = 0;
    while (line[i] ISNOT TERMINATOR)
    {
      temp[j] = line[i];
      i++;
      j++;
    }
    temp[j] = TERMINATOR;
    stcd_i(temp,&number2);
  }
 }
}
/* this subroutine checks the supplied character to see if it is a seperator,
   ie +,-,/ or *. If it is, then FALSE is returned, if not, then TRUE is
   returned.
*/
not_separator(c)
char c;
```


Select Vocabulary (commands 05, 06, 07 - level selection)

Commands 05, 06 and 07 are identical in format and are used to set the Levels 1, 2 and 3 respectively.

The Application may use as many levels as required. An example follows of a typical implementation method:

- Level 1 as user names, permanently resident. Receipt of a user name model would invoke a new user-specific vocabulary copy.
- Level 2 as user-specific models known to be permanently required during the current user's session.
- Level 3 as models which are to be activated/deactivated according to grammatical context or to new menu listings.

In addition a multi-file facility may be used. The application may make selections at all levels from up to 30 files provided that the number of active word models does not exceed 63 at any one time.

A call at a particular level deactivates all word models marked 0 in the mask, and activates all models marked 1, in the current 128-word block.
Only word models previously activated at this level will be deactivated by this command.

The Status word is returned zero if the command succeeds or one of the following is returned:

- 02 if the file does not exist or if there is no room for a new file.
- 03 if an attempt is made to activate more than 63 word models in total.

Programming examples

This section includes several versions of a simple integer calculator program.

The versions are in:

1. Microsoft C
2. Microsoft Compiled Basic
3. Microsoft interpretive Basic

The choice is provided to illustrate the techniques of interfacing which can vary considerably from language to language.

It should be noted that interpretive Basic is not an ideal language for Voice Applications. However it is a universal language which may be readily understood by the majority of computer users and serves to provide many with an introduction to voice technology.

In addition the example provided may easily be executed on the Apricot by Users who do not have compilers and assemblers. Those that do may find the interactive capabilities of interpretive Basic, together with the interfaces provided, useful for experimentation prior to preparing Applications in compiled form.

The interface and sequence of statements are unnecessarily complicated by inherent factors within the language. However, readers requiring an overview of the Application should be able to read the example and ignore the underlying problems.

The compiled versions presented are kept as simple as possible and do not attempt to do anything more than illustrate the use of commands and the requirements of interfacing.

```
        case '/':
          result = number1 / number2;
        break;

        case '*':
          result = number1 * number2;
          break;

        case FINISH:
          break;

        case HELP:
          printf("/nHelp Message/n");
          break;
      }
    if (operand IS FINISH) break;
    if (operand IS NOT HELP) printf("Result is %d/n",result);
    }
}
/*
   This subroutine outputs a prompt then switches the voice system on and
   waits for input. Once a line has been entered, either by voice or by
   keyboard, (the program can not tell the difference), the voice system
   is switched off and control returns to the calling program
*/

ginput()

{

   printf("/nPlease enter your calculation/n");

   viface.command = 4;                    /* switch Voice input device on */
   viface.iflag = 1;
   voice(&viface);

   scanf("%s",&line);                     /* get input line */

   viface.iflag = 0;                      /* switch Voice input device off */
   voice(&viface);
}

/* This subroutine takes the input line and checks the first character to see
    if it is either '?' - help, or 'E' - finish. If it is not, it will then
    separate out the first number, the operand and the second number.*/

decode()

{
char temp[30];
int i,j;
```

```
viface.command = 1                  /* init voice driver and validate vocabulary
strcpy(viface.filename,VOCNAME);
viface.offset = 0;
for (i=0;i<16;i++)
{
  viface.mask[i] = 0;
}
viface.iflag = 2;
viface.instance = 2;
viface.vptr = &vcmds;
voice(&viface);

show_□status                        /* test the returned status byte */

viface.command = 5;                 /* set up level 1 recognition */
strcpy(viface.filename,VOCNAME);
viface.offset = 0;
viface.mask[0] = 7;
for (i=1;i<16;i++)
{
  viface.mask[i] = 0;
}
voice(&viface);

show_status();

viface_command = 6;                 /* set up level 2 recognition */
strcpy(viface.filename,VOCNAME);
viface.offset = 0;
viface.mask[0] = 248;
viface.mask[1] = 255;
viface.mask[2] = 3;
for (i=3;i<16;i++)
{
  viface.mask[i] = 0;
}
voice(&viface);

show_status();

                                    /* enter the main loop */
for EVER
{
  ginput();                         /* read in a line */

  decode();                         /* check for '?' or 'E' else break the line
                                       down into first number, operand and
                                       second number*/


switch(operand)/*                     * perform the calculation */
{
  case '+':
    result = number1 + number2;
    break;

  case '-':
    result = number1 - number2;
    break;
```


Example 1: Microsoft C

The program below serves to provide a straight forward introduction to the application of Voice.

A file called CALCULAT.VOC must be created and trained prior to it's use with the following entries:

Entry	Prompt	Command?	Command
1	help	Y	? ∧ M
2	end	Y	E ∧ M
3	ok	Y	
4	back	Y	∧ H ∧ M
5	one	Y	1 ∧ M
6	two	Y	2 ∧ M
7	three	Y	3 ∧ M
8	four	Y	4 ∧ M
9	five	Y	5 ∧ M
10	six	Y	6 ∧ M
11	seven	Y	7 ∧ M
12	eight	Y	8 ∧ M
13	nine	Y	9 ∧ M
14	zero	Y	0 ∧ M
15	plus	Y	+ ∧ M
16	minus	Y	— ∧ M
17	multiply_by	Y	* ∧ M
18	divided_by	Y	/ ∧ M

The first three entries are designated by the Application as level 1. The remainder are designated level 2.

All word models are active throughout the execution of the program.

A machine code interface must be linked to the program before it can be used. It's purpose is to extract pointer information from the stack and set up the internal registers prior to calling the VIS via interrupt F6 hex.

A listing of the routine is provided on the following pages.

Voice interface

```
                PAGE    60,132
                TITLE   Voice Interface Program
;
                ;An interface program for applications written in
                ;languages that pass pointers to data structures
                ;as a single word offset on the system stack.
                ;
PGROUP  GROUP   PROG

PROG    SEGMENT BYTE    PUBLIC  'PROG'
        ASSUME  CS:PGROUP

        PUBLIC  VOICE
VOICE   PROC    NEAR

                ;This routine will be called with the following
                ;stack contents:
;               STACK   : param. block offset : +2
;               (SP) - >: return address      : 0

                PUSH    BP                      ;save working registers
                PUSH    SI
                MOV     BP,SP                   ;point to stack.(param.block offset
                                                ;will now be 4 bytes down the stack)
                MOV     SI,6[BP]                ;DS:SI will now point to the parameter block

                INT     0F6h                    ;call the VOICE programs

                ;No action is required on return from VOICE
                ;except to clear the param. block offset from the
                ;stack and return to the calling program.

                POP     SI                      ;restore the working registers
                POP     BP
                RET     2

VOICE   ENDP
PROG    ENDS
END
```


Microsoft C - simple integer calculator

This is a sample program written in Microsoft 'C' to show an application may use the ACT Voice system. This particular application is a simple integer calculator.

```
#include <stdio.h>                      /* standard 'C' definitions file*/

struct vparams                          /* the parameter block for voice commands */
{
  int command;                          /* the voice command*/
  int filename[5];                      /* the name of the current vocabulary file*/
  int offset;
  char mask[16];
  int iflag;
  char text[16];
  int instance;
  int vptr;
  char otext[36];
  int status;
  int indicator;
  int nowms;
  int nowmsact;
};

struct vparams viface;                  /* defines variable viface of type vparams*/

char line[30],operand;                  /* defines global variables */
int number1,number2;

#define VOCNAME "CALCULAT"              /* name of vocabulary file to be used */
#define VCOMM1 "? ^ M                       "
#define VCOMM2 "E ^ M                       "

/* define constants used by program */

#define SPACE ' '
#define FINISH 'E'
#define HELP '?'
#define TERMINATOR '/0'                 /*string terminator used by 'C' */

#define EVER (;;)                       /* causes an indefinite loop */
#define IS ==
#define ISNOT !=
#define TRUE 1
#define FALSE 0
#define OR¦ ¦

main()

{

                                        /* Define the local variables */

char vcmds[32];                         /* Vocabulary validation string */
int i;                                  /* work variable*/

  int result;                           /* the result of the calculation*/

  strcpy(&vcmds[0],VCOMM1);
  strcpy(&vcmds[16],VCOMM2);
```